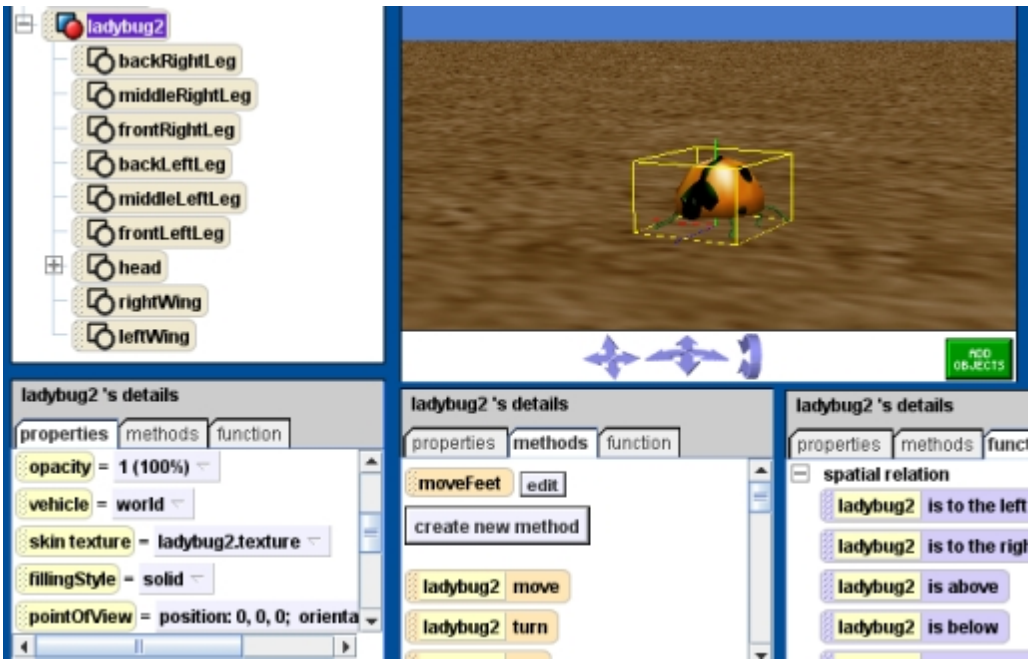
INTRODUCTION À POO



Baccalauréat S
Informatique et Sciences du Numérique

- 4.3 : Langages de programmation : POO

Référentiel de formation

Objectifs

A la fin de la séquence, l'élève doit être capable

- de définir ce qui distingue une programmation classique d'une programmation objet

Prérequis

- Introduction à la programmation
- Algorithmique
- Programmation

Programmation procédurale

Définition

Avec des langages tels que le C ou le Pascal, la résolution d'un problème informatique passe généralement par l'analyse descendante qui consiste à décomposer un problème en sous-problème jusqu'à descendre à des actions primitives.
On décompose ainsi un programme en un ensemble de sous-programmes appelés ***procédures*** qui coopèrent pour la résolution d'un problème.

Les procédures et fonctions sont généralement des outils qui produisent et/ou modifient des données.

Ainsi :

PROGRAMME = ALGORITHME + DONNEES

```
using namespace std;
int n;
int quitter()
{
    cout << "Appuyer sur la touche ENTREE pour finir!" << endl;
    return 0;
}
//------------------------------------------------------------
void ecrire_premiere_ligne(int cote)
{
    cout<<"\xDA";
    for (int i=1;i<cote-1;i++)cout<<"\xc4";
    cout<<"\xBF"<<endl;
}
//------------------------------------------------------------
void ecrire_ligne_intermediaire(int cote)
{
    cout<<"\xB3";
    for (int i=1;i<cote-1;i++)cout<<" ";
    cout<<"\xB3"<<endl;

}
//------------------------------------------------------------
void ecrire_derniere_ligne(int cote)
{
    cout<<"\xC0";
    for (int i=1;i<cote-1;i++)cout<<"\xc4";
    cout<<"\xD9"<<endl;
}
//------------------------------------------------------------
int main()
{
    cout<<"Donner une valeur sup\x82rieure \x85"<< " 0 : ";
    cin>>n;
    ecrire_premiere_ligne(n);
    for (int i=1;i<n-1;i++)ecrire_ligne_intermediaire(n);
    ecrire_derniere_ligne(n);
    aller_a_la_ligne();
    cout<<endl;
    //------------------------------------------------------------
    quitter();
}
```

Inconvénients

- L'évolution d'une application développée suivant ce modèle n'est pas évidente car la moindre modification des structures de données d'un programme conduit à la révision de toutes les procédures manipulant ces données.
- Pour de très grosses applications, le développement peut être très long

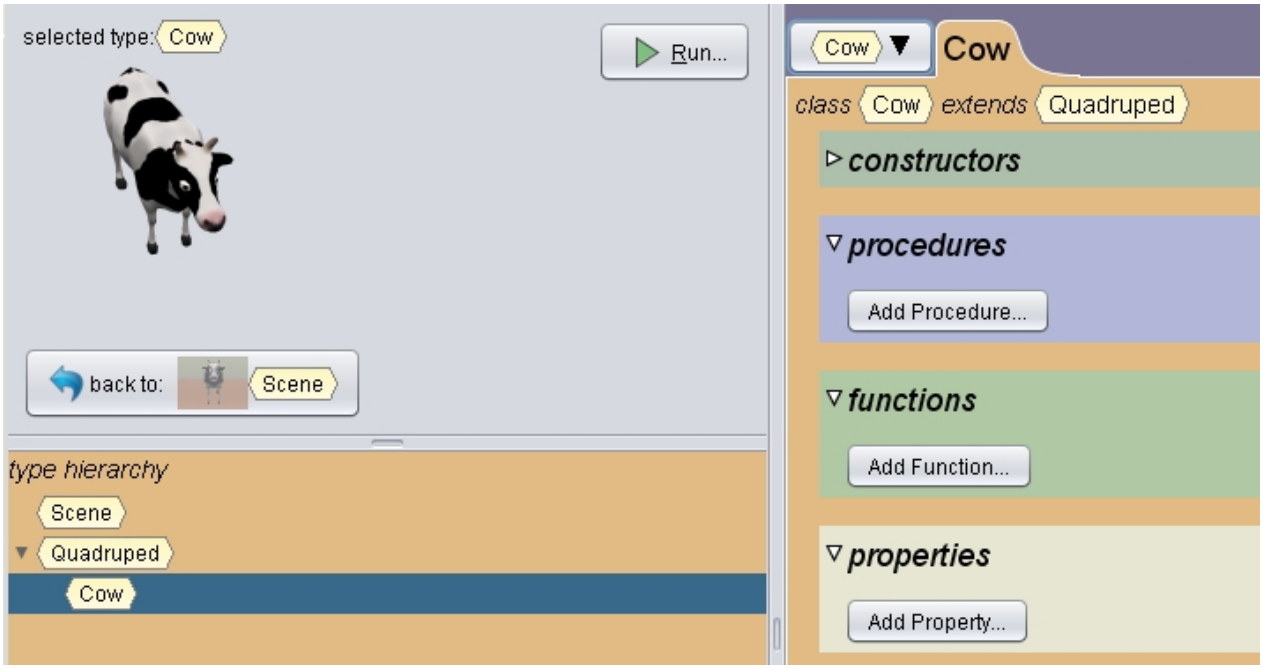
Programmation Orientée Objet

Définition

Les inconvénients de la programmation procédurales et en particulier les temps de programmation trop longs ont contribué à développer une nouvelle méthode de programmation appelée ***Programmation Orientée Objet*** (POO).

Le concept objet logiciel est né du besoin de modéliser des objets du monde réel.

Tout objet du mode réel peut être représenter par un objet, qu'il s'agisse d'un objet vivant, ou d'un objet construit.



Par exemple, prenons une vache...(Pourquoi pas ?).
- La vache possède plusieurs comportements : marcher, meugler, se coucher, ...
- Elle possède des caractéristiques propres : L'aspect de sa fourrure, son poids, taille, son nom, ...

Pour représenter cet animal dans l'approche objet :

- Les comportements seront représentés par des ***Procédures***, ***Méthodes*** ou des ***Fonctions*** (les noms varient selon les langages de programmation ou les auteurs).
- Les caractéristiques seront représentées par des ***Propriétés***.

Objet

Un objet est un "paquet" logiciel qui contient une série de ***Procédures*** (ou ***Méthodes*** ou ***Fonctions***) et de ***Propriétés***.

L'approche objet consiste à mettre en relation directement les données avec les algorithmes qui les
manipulent. Un objet regroupe à la fois des données et des algorithmes de traitement de ces données. Au sein d'un objet, les algorithmes sont généralement appelés des ***méthodes***, les données des ***propriétés***, ou des ***attributs***.

OBJET = PROPRIETES + PROCEDURES

En informatique, un objet est une entité virtuelle possédant

- des attributs (properties)
- des méthodes (methods)

Exemple : un point (un pixel) sur un écran est un objet.

Ses attributs peuvent être :

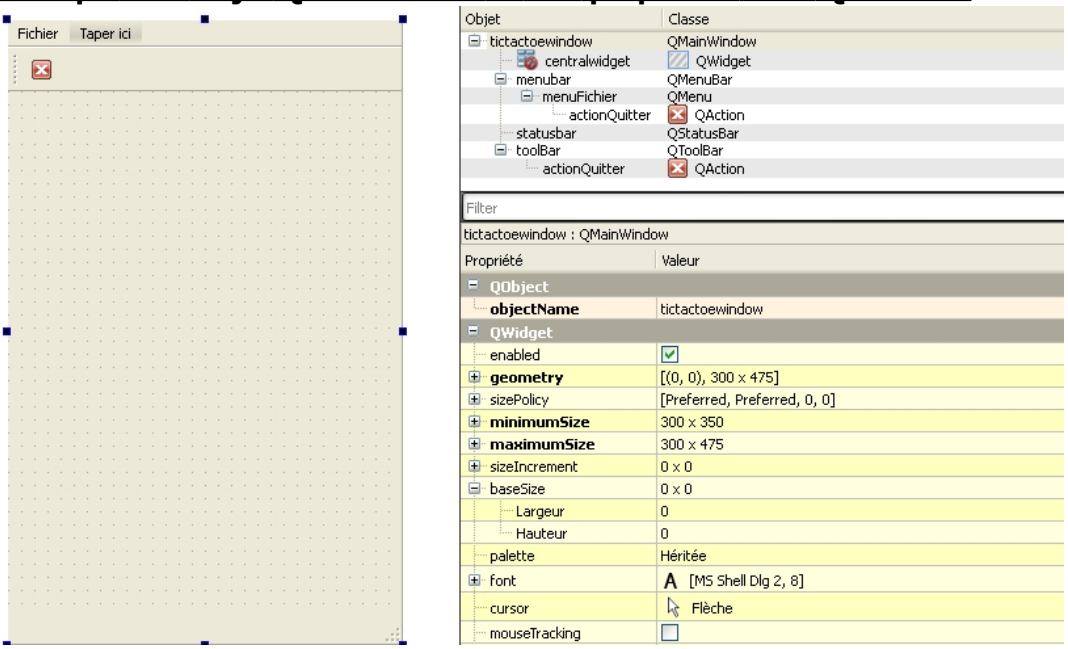
- Sa position horizontale
- Sa position verticale
- Sa couleur
- ....

Ses méthodes :

- positionner en x,y
- Afficher le point

- ....

Exemple d'un objet QMainWindow et ses propriétés sous QTCreator



Fondements de la POO

CLASSE (Class)

Une classe est un moule d'objet. Elle décrit les attributs et les méthodes de toute une famille d'objets.

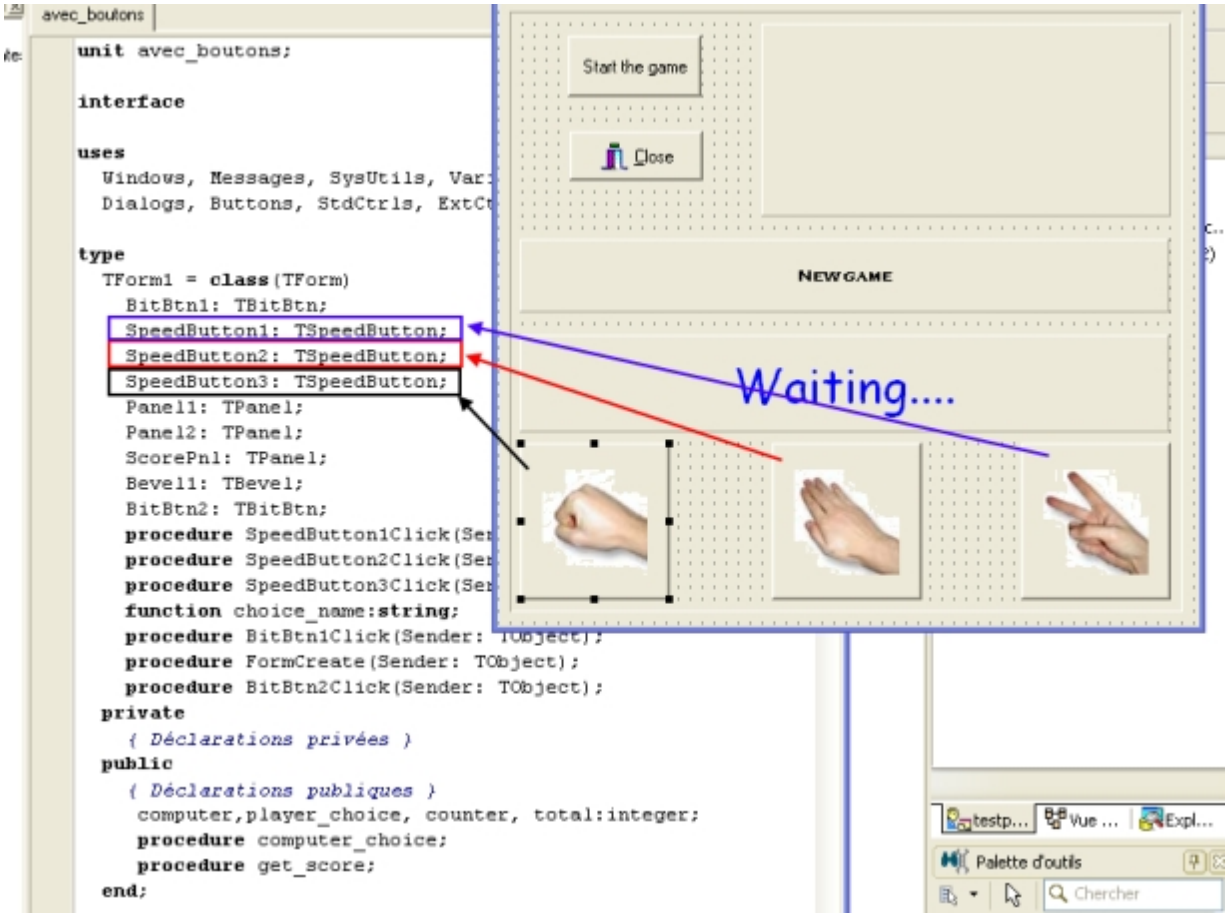
Une classe définit en particulier un type d'objet.

Les objets d'une même classe auront des attributs de même type, mais avec des valeurs différentes.

Une classe est la description générale d'une famille d'objet et un objet est une instance de classe. On parle d'instanciation.

Exemple :

Dans ce programme qui comporte 3 boutons, TSpeedButton est une classe de bouton comportant une image.
SpeedButton1, SpeedButton2 et SpeedButton3 sont des instances de la classe TSpeedButton.
L'image est une propriété de la classe et on remarque que chaque instance à une image différente.



Le polymorphisme

Le mot ***Polymorphisme*** vient du grec et veut dire de formes différentes.

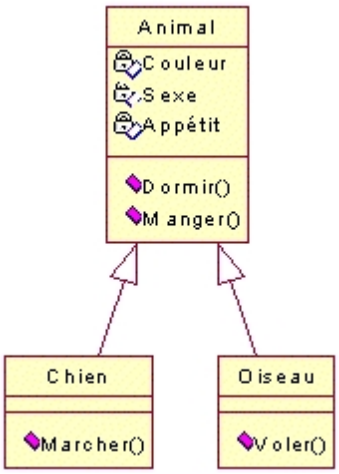
Le polymorphisme permet d'avoir des méthodes (polymorphisme ad-hoc), des attributs (polymorphisme paramétrique) de même nom, avec des fonctionnalités similaires, dans des classes sans aucun rapport entre elles (si ce n'est bien sûr d'être des filles de la classe objet).

Par exemple, la classe texte et la classe image dans un logiciel de traitement de texte peuvent avoir chacune une méthode "afficher". Cette méthode à la même action alors qu'elle manipule des objets différents.

L'héritage est une forme particulière de polymorphisme

L'héritage (Herited)

L'héritage est un mécanisme par lequel on définit une classe d'objet comme étant un cas particulier d'une classe plus générale.



Les classes peuvent être imbriquées à l'infini, et l'héritage s'accumule automatiquement. Cette structure est arborescente et s'appelle une hiérarchie de classe.
C'est grâce à l'héritage que l'on obtient une réelle "réutilisabilité" du code, une maintenance facilité et surtout une grande évolutivité. Toutes les "classes enfants" héritent des fonctionnalités écrite dans la "classe parent".

Exemple

Soit l'objet animal possédant les propriétés ***COULEUR, SEXE, APPETIT*** et les méthodes ***DORMIR()*** et ***MANGER().***
Les objets CHIEN et OISEAU héritent d'ANIMAL. Ils possèdent donc ses propriétés et ses méthodes qu'il ne faudra pas redéfinir, mais en plus CHIEN possède la méthode ***MARCHER()*** et OISEAU la méthode ***VOLER().***

Encapsulation

L'encapsulation est un mécanisme consistant à rassembler les données et les méthodes au sein d'une structure en cachant l'implémentation de l'objet, c'est-à-dire en empêchant l'accès aux données et aux codes informatiques par un autre moyen que les services proposés.

L'encapsulation permet donc de garantir l'intégrité des données contenues dans l'objet.

Pour ce faire, lors de l'écriture des classes on dispose de trois niveaux de lisibilité :

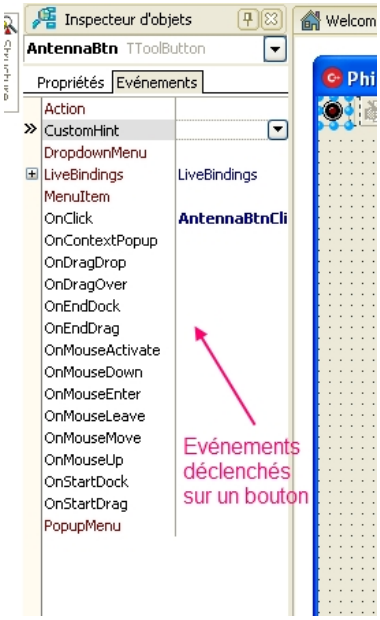
- **publique:** les fonctions de toutes les classes peuvent accéder aux données ou aux méthodes d'une classe définie avec le niveau de visibilité public.
- **privée:** l'accès aux données est limité aux méthodes de la classe elle-même. Il s'agit du niveau de protection des données le plus élevé
- **protégée:** l'accès aux données est réservé aux fonctions des classes héritières, c'est-à-dire par les fonctions membres de la classe ainsi que des classes dérivées

QT Creator :

mainwindow.h | <Selectionner un symbole>

```
1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5
6  namespace Ui {
7  class MainWindow;
8  }                          MainWindow hérite de QMainWindow
9
10 class MainWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:                                          Déclaration
15     explicit MainWindow(QWidget *parent = 0);    publique
16     ~MainWindow();
17
18 public slots:
19    void on_SetBtn_clicked();
20    void on_ResetBtn_clicked();
21
22 private:                                         Déclaration
23     Ui::MainWindow *ui;                          privée
24
25 };
26
27 #endif // MAINWINDOW_H
28
```

Messages



En programmation objet, les objets s'animent grâce aux méthodes.

Les méthodes sont activées grâce à des messages qui leur sont envoyés.

Les messages peuvent provenir

- d'un élément extérieur : clavier, souris, etc....
- de mécanismes internes au programme orienté objet : temporisation, signal provenant d'un autre objet.....